a coprocessor for performing [vector type] operations on a plurality of components of one pixel of the graphics data; and

a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory.

(Unchanged) 2. The graphics accelerator of claim 1 wherein the memory includes a data SRAM.

(Canceled) 3. The graphics accelerator of claim 1 further comprising a direct memory access (DMA) engine for loading the graphics data from memory through loading operations and transferring processed graphics data to the memory through storing operations.

(Unchanged) 4. The graphics accelerator of claim 1 wherein the coprocessor processes the plurality of components of each pixel in parallel as three elements of a vector.

(Unchanged) 5. The graphics accelerator of claim 4 wherein the plurality of components of each pixel comprise R, G and B components of RGB formatted graphics data.

(Unchanged) 6. The graphics accelerator of claim 5 wherein the pixels are in an RGB16 format.

(Unchanged) 7. The graphics accelerator of claim 6 wherein the R component has five bits, the G component has six bits and the B component has 5 bits.

(Unchanged) 8. The graphics accelerator of claim 6 wherein the graphics data is organized into 32-bit words, and each 32-bit word includes two pixels having RGB16 format.

(Unchanged) 9. The graphics accelerator of claim 8 wherein the two pixels are respectively selected by two special load instructions.

(Unchanged) 10. The graphics accelerator of claim 9 wherein the two special load instructions are for loading a left one and a right one of the two pixels, respectively.

(Unchanged) 11. The graphics accelerator of claim 7 wherein the coprocessor comprises an input register.

(Unchanged) 12. The graphics accelerator of claim 11 wherein the R, G and B components are expanded into 8-bit components through zero expansion when loaded into the input register.

(Unchanged) 13. The graphics accelerator of claim 4 wherein the plurality of components of each pixel comprise Y, U and V components of YUV formatted graphics data, and

wherein the Y, U and V components are also referred to as Y, Cr and Cb components, respectively, of YCrCb formatted graphics data.

(Unchanged) 14. The graphics accelerator of claim 13 wherein the pixels are in a YUV 4:2:2 format.

(Unchanged) 15. The graphics accelerator of claim 14 wherein the pixels are organized into 32-bit words and each 32-bit word contains two pixels.

(Unchanged) 16. The graphics accelerator of claim 15 wherein the two pixels in each 32-bit word is organized in a YUYV format, each of the first Y component, the U component, the second Y

component, and the V component occupies eight bits, a first one of the two pixels is comprised of a first Y component, the U component and the V component, and a second one of the two pixels is comprised of the second Y component, the U component and the V component.

(Unchanged) 17. The graphics accelerator of claim 15 wherein the two pixels are respectively selected by two special load instructions.

(Unchanged) 18. The graphics accelerator of claim 17 wherein the two special load instructions are for extracting a first one and a second one of the two pixels, respectively.

(Unchanged) 19. The graphics accelerator of claim 4 wherein the coprocessor has an instruction set that includes a special instruction for comparing between each element of a pair of 3-element vectors.

(Unchanged) 20. The graphics accelerator of claim 19 wherein the coprocessor further comprises a result register, and results of the three comparisons are stored in the result register.

(Unchanged) 21. The graphics accelerator of claim 20 wherein the results of the three comparisons are used together during a single conditional branch operation.

(Unchanged) 22. The graphics accelerator of claim 19 wherein the special instruction is for a greater-than-or-equal-to operation.

(Amended) 23. The graphics accelerator of claim 3 wherein the DMA engine moves data between the local memory and [an] the

Sub C4 B2 Cont

external memory at the same time the graphics accelerator is using the memory for its load and store operations.

(Unchanged) 24. The graphics accelerator of claim 23 wherein the external memory is a unified memory that is shared by a graphics display system, a CPU and other peripheral devices.

(Unchanged) 25. The graphics accelerator of claim 3 wherein the DMA engine includes a queue to hold a plurality of DMA commands.

(Unchanged) 26. The graphics accelerator of claim 25 wherein the plurality of DMA commands are executed in the order they are received.

(Unchanged) 27. The graphics accelerator of claim 25 wherein the queue comprises a mechanism that allows the graphics accelerator to determine when all the DMA commands have been completed.

(Unchanged) 28. The graphics accelerator of claim 25 wherein the queue is four deep for storing up to four DMA commands.

(Unchanged) 29. The graphics accelerator of claim 3 wherein the graphics accelerator is working on operands and producing outputs for one set of pixels, while the DMA engine is bringing in operands for a future set of pixel operations.

(Amended) 30. A method of processing graphics comprising the steps of:

loading a block of graphics data from main memory into local memory of a graphics accelerator having a processor and a coprocessor, the graphics data including pixels, each pixel having a plurality of components; [and]

performing [vector type] operations on the plurality of components of each pixel of graphics data using the coprocessor; and

concurrently transferring blocks of unprocessed data and processed data between the main memory and the local memory while the block of graphics data is being processed.

(Unchanged) 31. The method of processing graphics of claim 30 wherein the plurality of components comprises R, G and B components of RGB formatted graphics data.

(Unchanged) 32. The method of processing graphics of claim 31 wherein the pixels of the graphics data are in an RGB16 format.

(Unchanged) 33. The method of processing graphics of claim 32 further comprising the step of organizing the graphics data into 32-bit words, wherein each 32-bit word includes two pixels having RGB16 format.

(Unchanged) 34. The method of processing graphics of claim 33 further comprising the step of selecting each of the two pixels with one of two special load instructions.

(Unchanged) 35. The method of processing graphics of claim 34 wherein the step of selecting each of the two pixels comprises loading a left one of the two pixels.

(Unchanged) 36. The method of processing graphics of claim 34 wherein the step of selecting each of the two pixels comprises loading a right one of the two pixels.

(Unchanged) 37. The method of processing graphics of claim 30 wherein each of the plurality of pixels of graphics data comprises Y, U and V components of YUV formatted graphics data.

(Unchanged) 38. The method of processing graphics of claim 37 wherein the pixels are in a YUV 4:2:2 format.

(Unchanged) 39. The method of processing graphics of claim 38 further comprising the step of organizing the pixels into 32-bit words, wherein each 32-bit word contains two pixels.

(Unchanged) 40. The method of processing graphics of claim 39 wherein the step of organizing the pixels into 32-bit words comprises organizing each of the two pixels into a YUYV format, wherein each of the first Y component, the U component, the second Y component and the V component occupies eight bits, a first one of the two pixels is comprised of a first Y component, the U component and the V component, and a second one of the two pixels is comprised of the second Y component, the U component and the V component.

(Unchanged) 41. The method of processing graphics of claim 40 further comprising the step of selecting each of the two pixels with one of two special load instructions.

(Unchanged) 42. The method of processing graphics of claim 41 wherein the step of selecting each of the two pixels comprises loading the first one of the two pixels.

(Unchanged) 43. The method of processing graphics of claim 42 wherein the step of selecting each of the two pixels comprises loading the second one of the two pixels.

(Unchanged) 44. The method of processing graphics of claim 30 further comprising the step of comparing between each element of a pair of 3-element vectors, wherein each element of the 3-element vector is one of three components of each pixel.

(Unchanged) 45. The method of processing graphics of claim 44 wherein the three components of each pixel are R, G and B components.

(Unchanged) 46. The method of processing graphics of claim 44 wherein the three components of each pixel are Y, U and V components.

(Unchanged) 47. The method of processing graphics of claim 44 wherein the coprocessor includes a result register, and the method further comprising the step of storing results of the three comparisons in the result register.

(Unchanged) 48. The method of processing graphics of claim 47 further comprising the step of performing a single conditional branch operation using the results of the three comparisons.

(Unchanged) 49. The method of processing graphics of claim 44 wherein the step of comparing between each element of a pair of 3-element vectors comprises the step of performing a greater-than-or-equal-to operation between each element of a pair of 3-element vectors.

B

B4

(Amended) 24 ~~50~~. The method of processing graphics of claim 4 ~~30~~ wherein the graphics accelerator includes [a] the local memory for loading the graphics data, the method further comprising the step of moving data between the local memory and [an] the external memory using a direct memory access (DMA) engine at the same time the graphics accelerator is using the local memory for its load and store operations.

(Amended) 25 ~~51~~. The method of processing graphics of claim 24 ~~50~~ wherein the local memory is a data SRAM.

(Unchanged) 52. The method of processing graphics of claim 50 wherein the external memory is a unified memory that is shared by a graphics display system, a CPU and other peripheral devices.

(Unchanged) 53. The method of processing graphics of claim 50 wherein the DMA engine includes a queue for holding a plurality of DMA commands.

(Unchanged) 54. The method of processing graphics of claim 53 further comprising the step of determining whether all the DMA commands have been completed.

(Unchanged) 55. The method of processing graphics of claim 53 further comprising the step of receiving the plurality of DMA commands.

(Unchanged) 56. The method of processing graphics of claim 55 further comprising the step of executing the plurality of DMA commands in the order they are received.

(Unchanged) 57. The method of processing graphics of claim 53 wherein the queue is four deep for storing up to four DMA commands.

138

B

(Unchanged) 58. The method of processing graphics of claim 30 further comprising the step of bringing in operands for a future set of pixel operations using a direct memory access (DMA) engine while the graphics accelerator is working on operands and producing outputs for one set of pixels.

Please add new claims 59-61 as follows:

(New) -- 59. The graphics accelerator of claim 1, wherein the DMA includes a queue for storing DMA commands, wherein the processor posts a plurality of commands to the queue and the DMA executes the commands concurrently with the operation of the processor. --

(New) -- 60. The graphics accelerator of claim 1 wherein the coprocessor incorporates load and store functions for unpacking graphics data into a plurality of components, processing each of the plurality of components, and converting the plurality of components into a format suitable for storage. -

(New) -- 61. A method for sequentially processing blocks of graphics data, the method comprising the steps of:

transferring a first block of unprocessed graphics data from main memory to on-board memory;

processing the first block of graphics data;

transferring a second block of unprocessed graphics data from the main memory to the on-board memory while the first block of graphics data is being processed; and

transferring a third block of processed graphics data from the on-board memory to the main memory while the first block of graphics data is being processed. --